CLMPTO

March 19, 2003

LST 9/16/04

1. A method of communicating across an operating system using a plurality of processes and a plurality of memory sources disposed within one or more processors, said method comprising the steps of:

detecting an event within the system;

extracting an initial process address from one of the memory sources to determine a location of an initial process in response to detecting the event;

extracting an initial data address from one of the memory sources to determine a location of initial data to be used in the initial process in response to detecting the event;

executing executable code of the initial process located at the initial process address; and

extracting a second process address from one of the memory sources to determine a location of a second process to execute prior to the completion of the execution of the executable code of the initial process.

2. A method as set forth in claim 1 further including the step of extracting a second data address from one of the memory sources to determine a location of second data to use in the second process prior to the completion of the execution of the executable code of the initial process.

3. A method as set forth in claim 2 wherein the step of extracting the second process address from one of the memory sources is further defined as extracting the second process address from the initial data at the initial data address.

4. A method as set forth in claim 3 wherein the step of extracting the second data address from one of the memory sources is further defined as extracting the second data address from the initial data at the initial data address.

5. A method as set forth in claim 4 further including the step of retrieving an initial data set from the initial data at the initial data address for manipulation during execution of the initial process.

6. A method as set forth in claim 5 wherein the step of detecting the event is further defined as retrieving an initial parameter block, the initial parameter block includes the initial process address and the initial data address.

7. A method as set forth in claim 6 further including the step of executing executable code of the second process located at the second process address with the second data located at the second data address.

8. A method as set forth in claim 7 further including the step of positioning the initial process and the second process in an execution order.

9. A method as set forth in claim 8 further including the step of extracting a final process address from one of the memory sources to determine a location of a final process to execute and further including the step of executing executable code of the final process located at the final process address to halt communication across the operating system until the system detects another event.

10. A method as set forth in claim 8 further including the steps of measuring a predetermined condition during execution of the second process and modifying at least one of the initial data address and the second data address to retrieve different data addresses during the execution of subsequent processes.

11. A method as set forth in claim 10 further including the steps of measuring an execution frequency of one of the processes and modifying a process address stored in a data address associated with the process such that a different process executes when a predetermined number of executions is measured.

12. A method as set forth in claim 7 further including the steps of positioning a plurality of processes in an execution order and establishing a process address and the data address for each of the plurality of processes.

13. A method as set forth in claim 12 further including the steps of manipulating the position of the processes such that execution order is modified and re-establishing the process address and the data address for each of the manipulated processes.

14. A method as set forth in claim 13 further including the step of displaying the execution order to a user such that the user can modify the position of the processes.

15. A method as set forth in claim 14 further including the step of displaying the memory source of each of the processors to the user such that the user can manipulate the position of the processes based upon each processor.

16. A method as set forth in claim 15 further including the step of displaying the processes to a user such that the user can modify the data located at a data address associated with the displayed processes.

17. A method as set forth in claim 16 further including the step of storing fixed processes in one of the memory sources wherein data associated with the fixed processes remains unchanged.

18. A method as set forth in claim 8 further including the step of modifying at least one of the initial process address and the second process address to a different process address to define a different execution order of the initial and second processes.

19. A method as set forth in claim 8 further including the steps of measuring a predetermined condition during execution of the second process and modifying at least one of the initial process address and the second process address to retrieve a different process address during the execution of subsequent processes.

20. A method as set forth in claim 19 wherein the step of modifying at least one of the initial process address and the second process address is further defined as modifying at least one of the initial process address and the second process address by extracting a different process address from one of the memory sources whereby the process at the different process address has executed.

21. A method as set forth in claim 19 wherein the step of modifying either one of the initial process address and the second process address is further defined as modifying at least one of the initial process address and the second process address by extracting a different process address from one of the memory sources whereby the process at the different process address is to be executed.

22. A method as set forth in claim 12 further including the step of establishing a sub-execution order of processes when the sub-execution order of processes may be modified without interrupting the execution order of the operating system.

23. A method as set forth in claim 22 further including the steps of implementing the sub-execution order by altering the second process address and the second data address located at the initial data address and writing the altered process address and the altered data address of the sub-execution order into the initial data address.

24. A method as set forth in claim 23 further including the steps of executing processes of the sub-execution order and returning to the execution order once each of the processes of the sub-execution order are executed.

25. A method as set forth in claim 24 wherein the step of returning to the execution order is further defined as writing the second process address and the second data address into one of the sub-execution data addresses.

26. A method as set forth in claim 25 further including the step of beginning the operating system at the beginning of the execution order in response to detecting a new event and implementing the sub-execution order into the execution order.

27. A method as set forth in claim 1 further including the steps of extracting a plurality of initial process addresses and a plurality of initial data addresses to define a plurality of execution orders and executing the plurality of execution orders at the same time.

28. A method as set forth in claim 2 wherein the steps of extracting the initial data address and extracting the second data address is further defined as extracting a plurality of initial data addresses and extracting a plurality of second data addresses.

29. A method as set forth in claim 2 further including the step of executing executable code for a process with a plurality of data located at a plurality of data addresses.

30. A method as set forth in claim 5 further including the step of processing the initial data set during execution of the executable code to define an initial processed data set.

31. A method as set forth in claim 30 further including the steps of extracting an initial processed data address from one of the memory sources and writing the initial processed data set to the initial processed data address.

32. A method as set forth in claim 31 further including the steps of retrieving the second data set from the second data located at the second data address and processing the second data set during execution of the executable code to define a second processed data set.

33. A method as set forth in claim 32 further including the steps of extracting a second processed data address from one of the memory sources and writing the second processed data set to the second processed data address.

34. A method as set forth in claim 33 wherein the step of extracting the second processed data address is further defined as extracting a plurality of processed data address and writing the second processed data set to the plurality of processed data addresses.

35. (Amended) A method of communicating across an operating system using a plurality of processes and a plurality of memory sources disposed within one or more processors, said method comprising the steps of:

detecting an event within the system;

extracting an initial process address from one of the memory sources to determine the location of an initial process in response to detecting the event;

extracting an initial data address from one of the memory sources to determine the location of initial data to be used in the initial process in response to detecting the event;

executing executable code of the initial process;

retrieving the initial data from one of the memory sources at the initial data address;

continuing execution of executable code of the initial process with the retrieved initial data to define an initial processed data set;

extracting an initial processed data address from one of the memory sources;

writing the initial processed data set to the initial processed data address;

extracting a second process address from one of the memory sources to determine the location of a second process to execute prior to the completion of the execution of the executable code of the initial process;

extracting a second data address from one of the memory sources to determine the location of second data to use in the second process;

executing executable code of the second process;

retrieving the second data from one of the memory source at the second data

address;

continuing execution of executable code of the second process with the retrieved second data to define a second processed data set;

extracting a second processed data address from one of the memory sources;

writing the second processed data set to the second processed data address;

extracting a final process address from one of the memory sources to determine the location of a final process to execute;

executing executable code of the final process to halt communication of the system until the system detects the event.